

Robotics and Computer Vision Project:

0101001001101111011000100
11011110111010001110011₂

Group 5

Mike Bershad

Yevsey Liokumovich

Joe Martin

May 2009



Table of Contents

Chapter 1 Abstract	1
Chapter 2 Introduction	2
2.1 Background	2
2.2 Objectives	2
2.3 System Overview	2
2.3.1 Chassis	3
2.3.2 Arm	3
2.3.3 External Computer	4
Chapter 3 Algorithms and Other Software	5
3.1 Scale Invariant Feature Transform (SIFT)	5
3.2 EigenFaces	5
3.3 MatLab	6
3.3.1 LEGO NXT Control	6
3.3.2 Video Input	6
Chapter 4 Mechanical Design	7
4.1 Robot Chassis	7
4.2 Arm	7
4.3 Camera	7
Chapter 5 Testing and Experiments	8
5.1 Face Recognition	8
5.1.1 Figures of Merit	8
5.1.2 Evaluation Procedure	8
5.1.3 Results	8
5.2 Target Recognition	9
5.2.1 Figures of Merit	9
5.2.2 Evaluation Procedure	10
5.2.3 Results	10
5.3 Whole System	11
5.3.1 Figures of Merit	11
5.3.2 Evaluation Procedure	11
5.3.3 Results	11
Chapter 6 Conclusions	13
6.1 System effectiveness	13
6.2 Future Work	13
Appendix A Code	15
Appendix B References	18

Chapter 1

Abstract

The purpose of this project was to create a robot and vision processing system based on the techniques learned in the Rutgers University Robotics and Computer Vision class. The primary goal was to create a system that would use visual cues as commands, and perform actions on a specific target based on these commands. While this project is a very general example, it provides a framework for the many more advanced functions possible with higher-end hardware.

Chapter 2

Introduction

This section provides a basic overview of this project, and general background information about the field of computer vision.

2.1 Background

The field of computer vision is very new, but is growing at a rapid pace. Even 10 years ago, real-time object recognition required relatively massive computing power. Existing algorithms could only compare an image to static, pre-trained images, requiring a fixed camera position and prior knowledge about the position and orientation of the tested object. This changed in 1999 with the publication of David Lowe's paper "Object recognition from local scale-invariant features." [1] Lowe's method was superior to previous algorithms because it relied on recognizing small similar features between images, instead of using the compute intensive method of comparing high-dimensional eigen vectors in feature space. Lowe's algorithm became the standard for appearance based object recognition, and only a few implementations of the high-dimensional eigen vector method remain.

One example of a remaining implementation is the EigenFaces algorithm developed by Sirovich and Kirby in 1987. [2] EigenFaces has remained a popular algorithm for processing and recognizing faces due to its ability to differentiate images containing faces from images of other objects, and its relative low-dimensional eigen vectors (achieved by optimizing the feature space used for the features common to all human faces). Both these features allow EigenFaces to run quickly, and remain a popular choice for human face recognition.

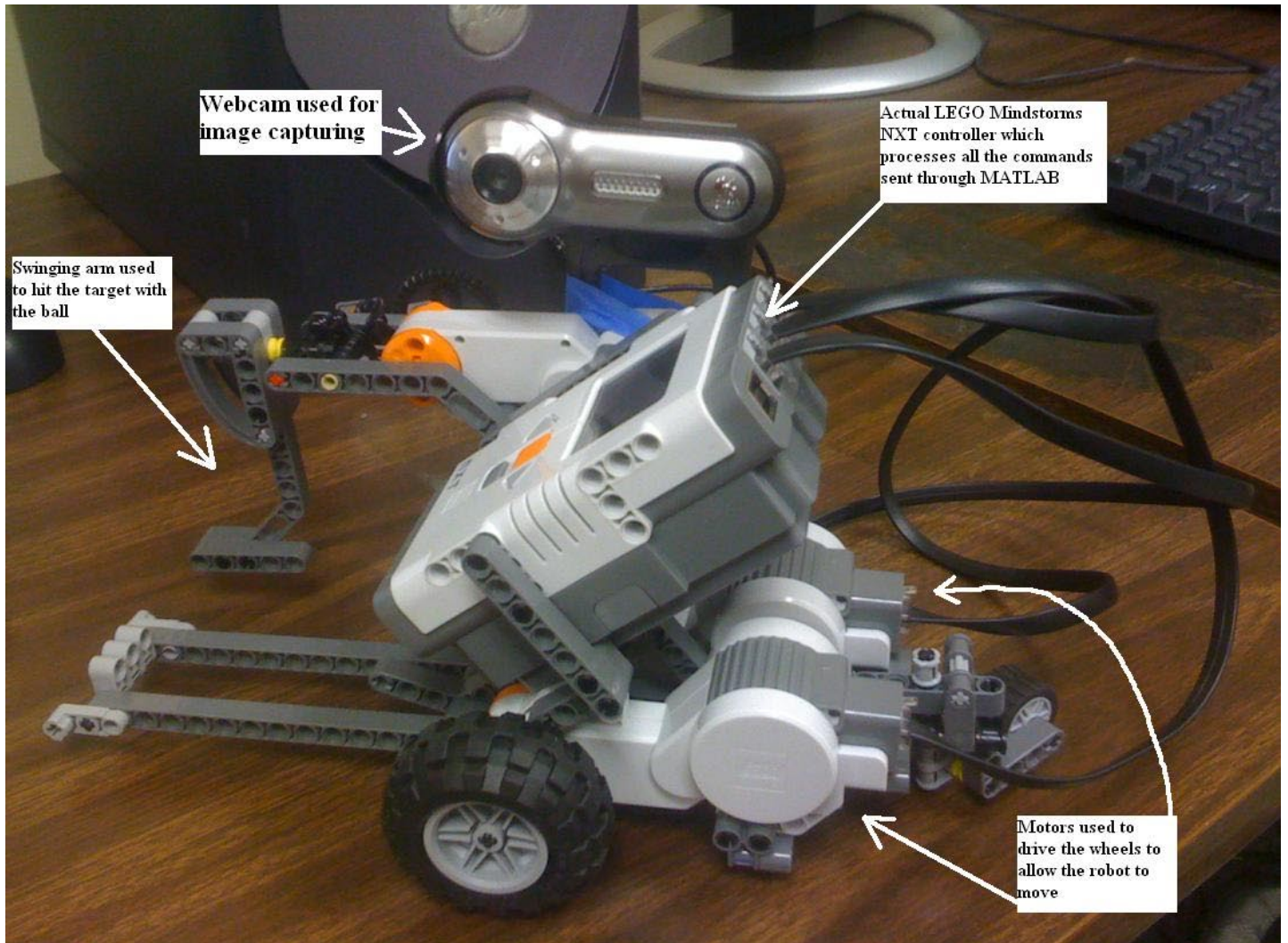
Both of these algorithms were evaluated as a part of this project, and conclusions regarding their effectiveness are presented below.

2.2 Objectives

- Build a robot capable of both movement and vision
- Implement a software interface to gather visual data from and control the movement of the robot
- Process visual data for visual cues
- Based on these cues, command the robot to perform a specific action on a specific target.

2.3 System Overview

Our system consists of three major components: a robot, consisting of a chassis and launcher arm, and an external computer used to process visual images.



2.3.1 Chassis

The chassis used on our robot is based on the standard one described in the LEGO Mindstorms NXT documentation. It has two main wheels, one on each side, with each driven directly by a NXT motor. There is also a third free wheel at the rear of the robot, providing stability. This configuration allows the robot to drive forward or in reverse, spin, or turn while moving. The NXT Brick is mounted above the motor housings, at a 45-degree angle to vertical. This makes its screen visible from above.

2.3.2 Arm

The arm's purpose is to strike a ball, launching it at a target. The arm is mounted on the side of the robot and rotates freely. Its speed can be varied, allowing the robot to control how hard it hits the ball, controlling how far the ball will go.

2.3.3 External Computer

According to LEGO, the Mindstorms NXT brick has relatively limited processing power (a 48 MHz Atmel ARM processor). [3] For this reason, an external computer was used to process image data gathered by the robot. The standard computer workstation in the Robotics and Computer Vision Lab at Rutgers University was chosen for reasons of convenience. This computer is a Dell with a Pentium 4 - 3.40 GHz processor and 512 MB of RAM. This computer uses MatLab as well as several associated libraries and algorithms to capture and process the images from the robot, and control the robot's movements.

Chapter 3

Algorithms and Other Software

3.1 Scale Invariant Feature Transform (SIFT)

SIFT is an algorithm in computer vision to detect and describe local features in images developed by David Lowe in 1999. The algorithm is widely used due to its robustness as it is invariant of scale or image rotation. The algorithm works by first convolving an image with a Gaussian filter at different scales, followed by taking the difference of each Gaussian blurred image. Keypoints in the images are found as the local maxima/minima of the difference of Gaussian (DoG) at different scales. A DoG image is found by the following equation:

$$D(x, y, \sigma) = L(x, y, k_i \sigma) - L(x, y, k_j \sigma) \quad [\text{Eq 1}]$$

where $L(x, y, k\sigma)$ is found by convolving the image $I(x, y)$ with a Gaussian filter at scale $k\sigma$. The mathematical equation can easily be seen below.

$$L(x, y, k\sigma) = I(x, y) * G(x, y, k\sigma) \quad [\text{Eq 2}]$$

The keypoints in the image are the local extrema of the DoG image and found by comparing each pixel in the DoG image to its eight neighboring pixels at the same scale and also to nine neighboring pixels at successive scales. If the pixel value is a max or min, it is considered a keypoint.

To obtain a higher accuracy in the SIFT algorithm, the low contrast keypoints are discarded. This is done by computing the second order Taylor expansion of $D(x)$ and if it's greater than a certain threshold, the keypoint is discarded. However, the DoG image still has a very high edge response which makes the keypoints unstable even at the slightest noise. These points are filtered out by comparing the sum of the derivatives in the x and y directions to the product of the same derivatives. If this ratio is higher than a threshold then the keypoints are placed along an edge and are rejected.

3.2 EigenFaces

The team decided to include face detection in the project, one such algorithm is called EigenFace. The algorithm requires a training set of images taken under the same lighting conditions and are normalized such that the eyes and mouths line up in all of the training set images. The mean is then calculated and subtracted from each of the training set images. The eigen values and eigenvectors are found from the covariance matrix with the eigenvectors called EigenFaces. The EigenFaces shows the directions in which the training set images differ from the mean image. Once the training stage is complete, the system is ready to perform recognition of a new image. The new image is decomposed into its eigen values and eigenvectors. The image is match to one of the training set images by minimizing the Euclidean distance between the

eigenvector of the training set and that of the new image. The algorithm was implemented in the project at first but was later abandoned due to its poor performance. The algorithm was highly dependant on the lighting conditions in the room which couldn't be controlled. Also, the training set had to be normalized such that all the eyes and mouths lined up which is hard to accomplish with a handheld web cam. The last reason to abandon EigenFaces was that the training set was relatively small and gave us false matches.

3.3 MatLab

3.3.1 LEGO NXT Control

The project was implemented in MATLAB. In order to communicate with the Lego NXT Robot, the “LEGO MINDSTORMS NXT Toolkit for MATLAB” was downloaded from the MathWorks website. The toolbox allowed direct control of the NXT Robot by sending motor control commands to the robot.

3.3.2 Video Input

The `videoinput` command from the image acquisition toolbox was used to acquire images from the supplied web cam. The `videoinput` command initializes a video object, with a desired resolution. The video stream is converted into grayscale by setting the returned color space property in the video object to grayscale. The `preview` command allows a live preview of the video stream. One frame of the video stream is captured by using the `getsnapshot` command

Chapter 4

Mechanical Design

The robot used in this project needed to achieve two goals: the ability to move, and the ability to perform some task. The task we chose was to launch a ball. This was chosen for simplicity's sake, and because the power with which the ball is launched can be easily varied. Movement was easily achieved by modifying the standard LEGO NXT "Tri-Bot" chassis. [4]

4.1 Robot Chassis

The chassis used on our robot is based on the standard "Tri-Bot" described in the LEGO Mindstorms NXT documentation. It has been modified, so that the forward gripper and sensor arm hard-point has been replaced by a trough. The trough holds a ball. The trough is sloped so that the ball rests at the bottom, in the proper position for the arm to launch the ball. The NXT Brick mount has also been modified to allow the arm to mount on the right side of the robot.

Two wheels drive the robot, one on each side. These wheels are directly attached to two NXT Motors. A third wheel mounted at the rear of the robot provides stability. The robot drives forward or backward by rotating both wheels in the given direction. If the wheels are rotated in opposing directions the robot spins in place, allowing for high maneuverability. Additionally, if one wheel is rotated faster than the other the robot will turn, allowing for efficient movement.

4.2 Arm

The arm consists of a standard NXT Motor with a frame extending from the main housing forward towards the rotor. This framework contains a set of bevel gears used to rotate the rotational drive axis by 90 degrees. This drives an arm, similar to a golf club, which rotates through an unlimited 360-degree range of motion. The arm strikes the ball, launching it to the left relative to the orientation of the robot. The arm strikes the ball at an upward angle, giving it enough loft to exit the trough, and travel about 8-12 inches before hitting the ground at maximum power (assuming the ground to be level).

4.3 Camera

The camera is mounted on the side of the arm's motor assembly. The camera's body includes a rubberized base with a deformable metal core. This base is shaped to fit snugly over a round protrusion on the motor casing, and is taped in place. The camera is oriented to look over the robot's left side, above the NXT Brick.

Chapter 5

Testing and Experiments

5.1 Face Recognition

Both SIFT and EigenFaces were evaluated for use as a face recognition algorithm. While EigenFaces was the faster and more efficient algorithm, it proved ineffective at matching images of faces taken from slightly different camera positions. Due to the movement of the camera between uses, this was a major problem for EigenFaces, and so SIFT was used for the final version. If, in a different situation, the relative positions of the face and camera could be fixed, EigenFaces would be the superior choice.

5.1.1 Figures of Merit

To test face recognition, the accuracy of both algorithms was evaluated multiple times, using faces in the database, unknown faces, and non-face objects. The main Figure of Merit used for comparison was the number of correct identifications during a set of trials.

5.1.2 Evaluation Procedure

To evaluate each algorithm, training images were created of several faces. These images were entered into each algorithm's respective database. Next, a set of test images was each given to the algorithm to identify in turn. The test images consisted of the training images, images of each of the training faces taken from slightly different angles, images of faces that were not in the training set, and images of non-face objects. The result of the identification was recorded as correct or not. The time required to reach a result was evaluated, but not recorded.

5.1.3 Results

The EigenFaces algorithm proved the fastest at identifying the faces from the training set, but incorrectly identified the same faces when viewed from a slightly different perspective. It also failed to determine that the unknown faces were in fact unknown. It did determine that non-face images were not faces.

SIFT was significantly slower than EigenFaces, but was far more accurate with its identifications. After varying the SIFT threshold values, the algorithm was able to correctly identify or exclude each test image. The ideal SIFT feature threshold value for face recognition was found to be 2.2, and the ideal minimum number of feature matches for a face match was found to be 12.

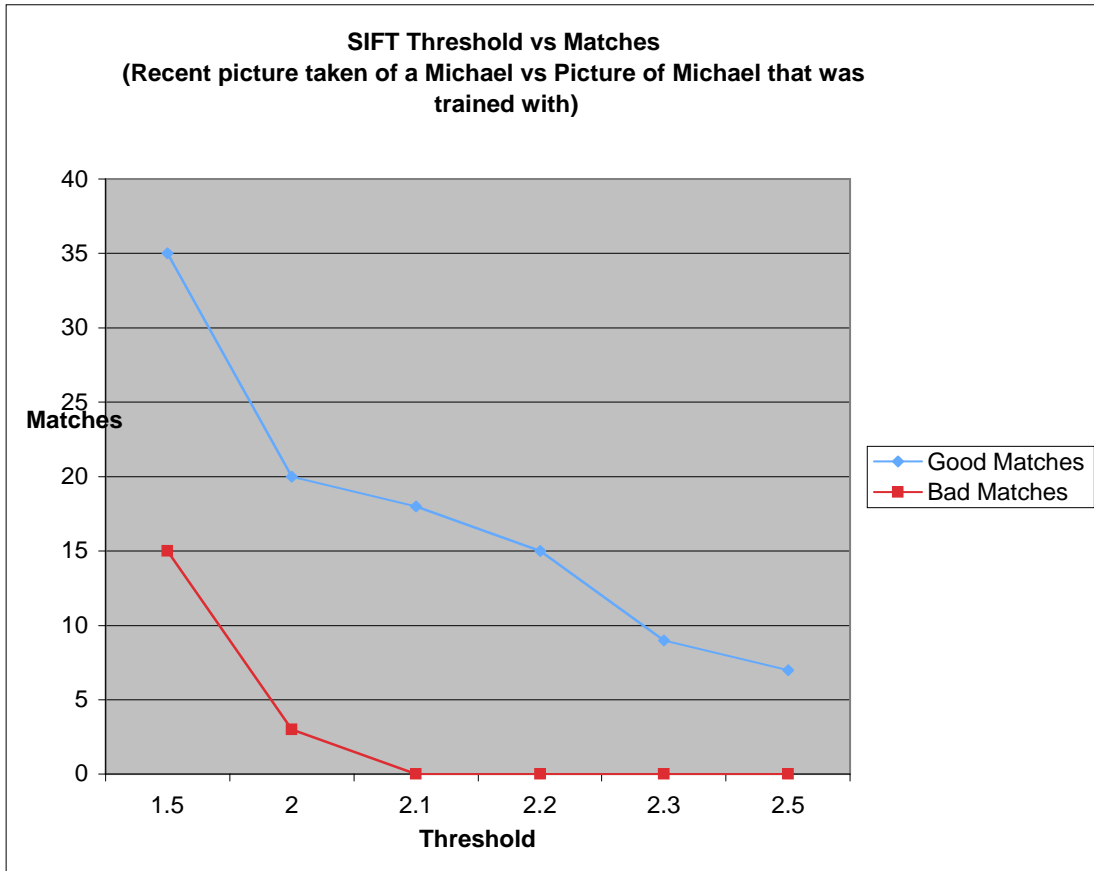


Figure 5-1: Good and Bad Feature Matches for Faces vs. SIFT Feature Threshold levels. 2.2 was chosen as an ideal feature threshold because it doesn't return any bad (extraneous) feature matches, while maximizing good (correct) feature matches.

Test	Matches
vs. Same Face	15
vs. Other Face	8
vs. Non-Face	3

Figure 5-2: Test results for SIFT face recognition. 12 was chosen as an ideal minimum number of feature matches to qualify as a face match because it is well above the number of feature matches returned for another face, while allowing for an image with fewer than average matches to still count as a correct identification.

5.2 Target Recognition

SIFT was again evaluated as a target recognition algorithm. As with face recognition, by varying the recognition parameters an ideal set of parameters was found.

5.2.1 Figures of Merit

Again, SIFT was evaluated by measuring the number of correct identifications for a given set of tests. These tests included images of the targets and images of non-targets.

5.2.2 Evaluation Procedure

The evaluation procedure for SIFT target recognition was very similar to that for SIFT face recognition. Training images were created for each target, and were entered into the SIFT database. A series of test images including the training images, images of the targets from slightly different perspectives, and images of non-targets was again given to the SIFT algorithm to identify. The result of the identification was recorded as correct or not.

5.2.3 Results

SIFT performed as well as with face detection, and was able to correctly identify or exclude every test image once the correct parameters were found. In this case, the ideal feature detection threshold was again found to be 2.2, and the ideal minimum number of feature matches for a target match was found to be 14.

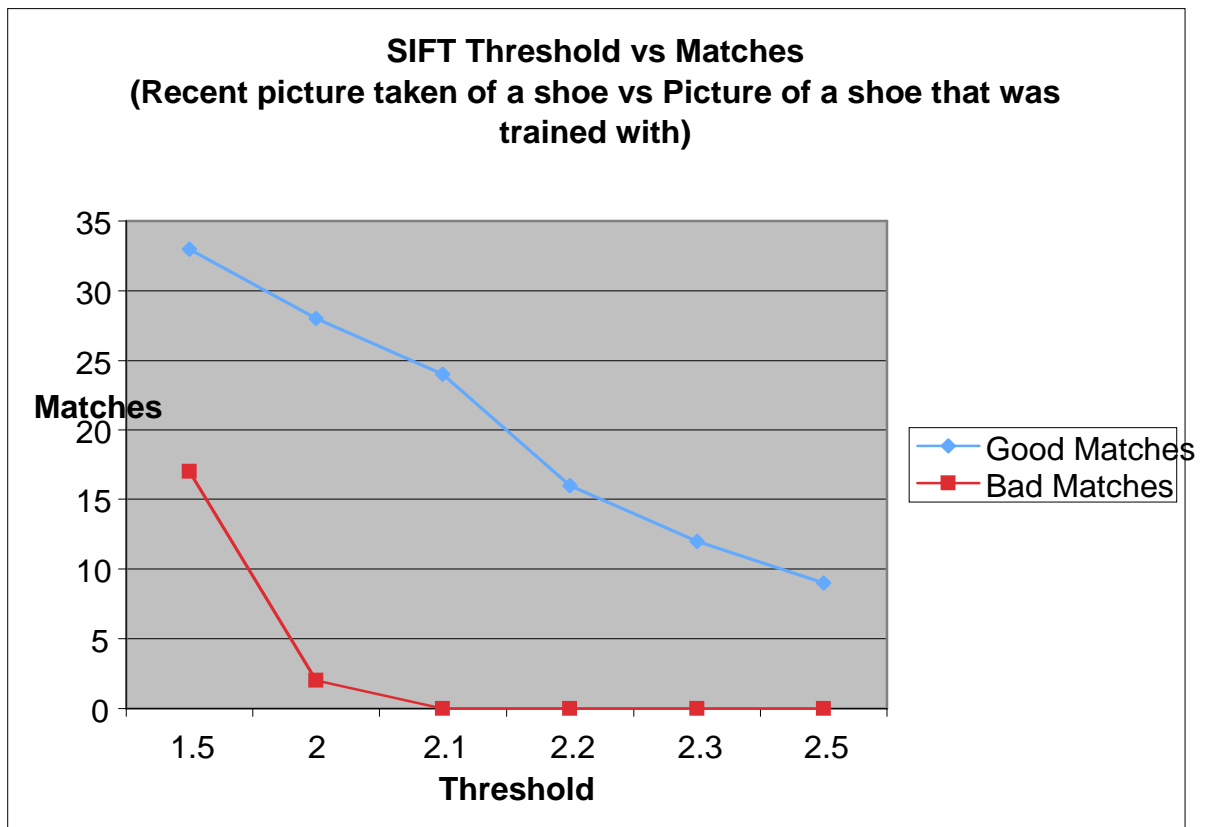


Figure 5-3: Good and Bad Feature Matches for Targets vs. SIFT Feature Threshold levels. Again, 2.2 was chosen as an ideal feature threshold because it doesn't return any bad (extraneous) feature matches, while maximizing good (correct) feature matches.

Test	Matches
vs. Same Target	16
vs. Other Target	9
vs. Non Target	3

Figure 5-4: Test results for SIFT target recognition. In this case, 14 was chosen as an ideal minimum number of feature matches to qualify as a face match because it is well above the number of feature matches returned for another target, while allowing for an image with fewer than average matches to still count as a correct identification.

5.3 Whole System

Reading about the separate aspects, it becomes clear that for the purposes specified for our project, using EigenFaces was out of the question due to the motion aspect of the task. Fortunately for our group, there was a SIFT command that helped us tremendously in recognizing who was in front of the camera, and which shoes were theirs. The aspect that became difficult was running various trials to establish the most effective threshold parameters.

5.3.1 Figures of Merit

To test the system as a whole, our group ran multiple tests comparing different faces to faces in our database and comparing faces that were not in the database to ones that were so make sure that our program was not incorrectly identifying people. Similarly, we ran multiple tests with various shoes, both in our database and ones that were not, and even objects, such as an oscilloscope, to make sure that we did not have any grossly large mistakes. After we made sure that the specific components worked, we tried running the program as a whole multiple times, ensuring that the right shoe was being targeted based on the picture taken of a person in the beginning.

5.3.2 Evaluation Procedure

To evaluate our program working as a whole, we compared how many times the robot correctly identified the shoe based on the picture taken of a person. To make sure that the system as a whole worked correctly, we took various pictures of the persons face with several lighting conditions; for the shoe finding aspect, we tried changing around the order of shoes and angling them slightly to account for “real-world” conditions. The results of the tests were recorded as correctly identified or not.

5.3.3 Results

After running the robot through various tests with various lighting conditions and placing the shoes and faces in different positions, we ended up having the robot correctly identify the person’s shoe that had their picture taken, if it was in our database, and state that there was no match found if the person was not previously in our database.

The best values for our SIFT algorithm, was a feature detection threshold of 2.2, as the default value of 1.5 allowed too many erroneous points detected as a match, where as the upper limit of 2.5 scrutinized the images too closely and did not allow for a proper identification.

Chapter 6

Conclusions

6.1 System effectiveness

In the early stages of the project when we were running various tests with variables, using the default value for SIFT threshold produced a match almost every time, even when the objects were not the same, including identifying Michael's face as Yevsey's so we knew that the default value wasn't exactly what we wanted to use. But after testing and varying the threshold between 1.5 (the default value) and 2.5 we found out that the best value was 2.2, which made the correct identifications and was not too harsh when trying to match photos.

After the many trial runs that we performed and after testing the effectiveness of various SIFT thresholds, we produced a fully working system that detected the correct person in front of the Webcam and correctly identified the corresponding shoe, given the right lighting conditions, 100% of the time.

6.2 Future Work

This project describes a framework for autonomous, visually-aware robots. Due to the hardware limitations of the LEGO NXT system (low processor power, limited inputs and outputs, etc.), only a very general example could be given. A more advanced implementation might feature a more robust chassis constructed of metal or plastic. It might utilize a more advanced camera (ie, one with higher resolution, or user controllable focal length, aperture, etc.). The biggest and easiest improvement, however, would be the use of a more capable processor. A system conforming to the PC-104+ standard would be ideal for this purpose. These machines are small, approximately 4" on a side, yet are powerful, with typical processor speeds ranging from 250 MHz to 1.5 GHz. These devices also are expandable with a wide range of peripherals, including interfaces for many standard busses (I²C, CAN, SSI, etc.) and general purpose I/O.

Improvements can also be made to the software used in this project. The most time-consuming task performed by the robot is the SIFT processing of potential targets. The EigenFaces algorithm was streamlined by optimizing the comparisons so as to be specific to the category of objects to be identified (ie, faces). It may be possible to similarly optimize the SIFT algorithm to identify targets of a particular type. The massive improvement that this optimization brought to EigenFaces suggests that similar efforts are warranted with SIFT.

Another possible improvement to SIFT would be filtering of the identified matching feature so as to exclude the few incorrect matches that are identified. This can be accomplished by comparing the relative position of each match within the image. When matching images with the same scale are viewed side-by-side with their features highlighted and lines connecting corresponding features, the correctly identified feature lines will be parallel. Furthermore, if the images have different

scales, the feature lines will converge to a single point (See Fig. 6-1, red lines). By removing features whose lines don't fit with this pattern (See Fig. 6-1, blue lines), extraneous feature matches can be filtered, resulting in a higher degree of accuracy.

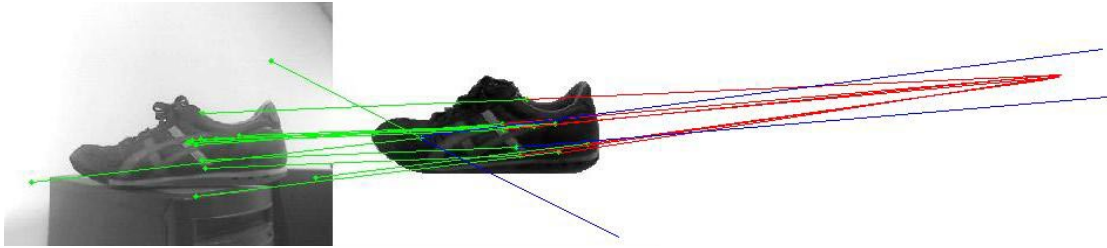


Figure 6-1: Illustration of feature line extensions. Red lines are valid feature matches, and converge to a point. Blue lines are extraneous feature matches, and do not converge.

Appendix A

Code

Below is our MatLab code:

```
COM_CloseNXT all
clear all
close all
handle = COM_OpenNXT('bluetooth.ini', 'check');
COM_SetDefaultNXT(handle);

StopMotor('all', 'off');
ResetMotorAngle(MOTOR_A);
ResetMotorAngle(MOTOR_B);
ResetMotorAngle(MOTOR_C);
% trainingImage=imread('\images\shoe_training.tif');
% trainingImage=rgb2gray(trainingImage);
vid=videoinput('winvideo',1, 'RGB24_320x240');
set(vid, 'ReturnedColorSpace', 'grayscale');
preview(vid)

reply=input('Press Enter when you are ready to have your picture
taken', 's');
if isempty(reply)
    faceTestImage=getsnapshot(vid);
    NXT_PlayTone(800,500);
end
[frame1, desc1]=sift(faceTestImage);

M=4;
faceThreshold=12;
j=1;
objectSelection=0;
while j~=0 && j<=M
    str=strcat('images\',int2str(j),'.TIF');
    eval('img=imread(str);');
    [frame2, desc2]=sift(img);
    match=siftmatch(desc1, desc2,2.2);
    plotmatches(faceTestImage, img, frame1, frame2, match)
    if length(match)>faceThreshold
        objectSelection=j;
        j=-1;
    end
    j=j+1;
end

fakevariable = 1;

%case 1 is Joe's shoe
%case 2 is Mike's shoe
%case 3 is Antonio's shoe
```

```

%case 4 is Yevsey's shoe
%case 5 is the oscilloscope
switch objectSelection
    case 0
        display('It appears that you have both of your shoes
on...');
        fakevariable = 0
    case 1
        trainingImage=imread('images\joeShoe_training.TIF');
    case 2
        trainingImage=imread('images\mikeShoe_training.TIF');
    case 3
        trainingImage=imread('images\antShoe_training.TIF');
    case 4
        trainingImage=imread('images\sevaShoe_training.TIF');
    case 5
        trainingImage=imread('images\oscope_training.TIF');
end

if (fakevariable == 1)
if ndims(trainingImage)==3
    trainingImage=rgb2gray(trainingImage);
end

[frame2, desc2]=sift(trainingImage);

i=1;
matchThreshold = 14;

while (i==1)
    NXT_SendKeepAlive('dontreply');

    pause(2);
    testImage=getsnapshot(vid);

    [frame1, desc1]=sift(testImage);
    match=siftmatch(desc1, desc2,2.2);
    plotmatches(testImage, trainingImage, frame1, frame2, match)
    size(match)

    if length(match)>=matchThreshold
        i=0;
        NXT_PlayTone(800,500);
        SetMotor (MOTOR_C)
            SetPower(-3)
            SetAngleLimit(30)
            SpeedRegulation on
            SendMotorSettings
        pause(2)
        SetPower (100)
        SetAngleLimit (360)
        SendMotorSettings
    end
end

```

```
SetMotor (MOTOR_A)
  SyncToMotor(MOTOR_B);
  SetPower (60)
  SetAngleLimit (180)
SendMotorSettings
WaitForMotor(MOTOR_A);
StopMotor('all', 'off')
```

```
end
end
```

```
COM_CloseNXT(handle);
```

Appendix B

References

- [1] Lowe, David G. (1999). "Object recognition from local scale-invariant features". *Proceedings of the International Conference on Computer Vision 2*: 1150–1157.
- [2] L. Sirovich and M. Kirby (1987). "Low-dimensional procedure for the characterization of human faces". *Journal of the Optical Society of America A* 4: 519–524.
- [3] The LEGO Group. (2006). "LEGO Mindstorms NXT Hardware Development Kit". The LEGO Group.
- [4] The LEGO Group. (2006). "LEGO Mindstorms Education" (Instruction Booklet). The LEGO Group.